University of Southern Denmark

DM852

Heuristics & Approximations Algorithms

Submitted To:
Marco Chiarandini
Lene Monrad Favrholdt
IMADA
Mathematics & Computer Science Department

Submitted By :
Alexander Lerche Falk
Kicito Narongrit Unwe
Spring - Master of Computer Science

Contents

1 Introduction

This project shows heuristics algorithms, which can resolve the Capacitated Vehicle Routing Problem (CVRP). If the heuristic can resolve the problem, we are going to apply Local Search algorithms to improve the results from the heuristics. The CVRP gives insight into how to deliver orders to customers using the shortest route for each vehicle, while preserving the capacity limit of each vehicle.
newline We have created two heuristic algorithms: one using a nearest-neighbour terminology and the other one taking inspiration of clustering close customers. We also show two self-implemented Local Search algorithms. Lastly, we are going to compare our own ideas against more common used algorithms, which are tested and proved to work.

— Alexander & Kicito

2 Custom Heuristics Algorithms

The first algorithm, which we have created, takes the approach of the Nearest Neighbour idea. Given an instance of CVRP, the starting point the depot where the vehicles are being loaded with customer requests is the point, where all calculations start. We start by checking the nearest point from the depot and add it to the first route. We then add the shortest path from the newly added point to the route and continue until we are out of capacity. Each point (customer) has a capacity, which adds up until the max is reached. We continue during the same from adding the first shortest route from the depot until we have reached our capacity limit. At one point, we have x amounts of routes, covering the problem instance.

Algorithm 1 Custom CVRP Heuristic - Nearest Neighbour Approach

```
1:  procedure MyProcedure
2:      stringlen ← length of string
3:      i ← patlen top:
4:      if i > stringlen then return false
5:      j ← patlen loop:
6:      if string(i) = path(j) then
7:          j ← j − 1.
8:          i ← i − 1.
9:          goto loop.
10:         close;
11:     i ← i + max(delta_1(string(i)), delta_2(j)).
12:     goto top.
```